How to use the OOTiA Handbook

What should we consider when we go down the OO path?

Prepared for the SW DER Conference
Norfolk, Virginia

July 27, 2005

How to Use the OOTiA Handbook

Material Source and Disclaimers

- The FAA is in the process of developing course material to address OO technologies and the OO Handbook – the bulk of this material has been selectively reused from that work

- RTCA/DO-178B and the OOTiA Handbook glossaries should be consulted for the definition of any technical terms that may be in question

- As a caveat, which you will see several times, the OOTiA Handbook is not guidance material – it was considered “best practice” at the time it was written and compiled

- SC-205 has an OO subcommittee looking at OO issues and considering the best way to deal with OO in terms of a DO-178C

How to Use the OOTiA Handbook

Everything you need to know on one slide…

- Volume 1 – Introduction
- Volume 2 – OO things to worry about (either avoid or have an answer)
- Volume 3 – Potential OO answers (not the only answers or all the answers)
- Volume 4 – The kinds of OO questions the certification authorities may ask

Questions?

How to Use the OOTiA Handbook

OOTiA Handbook Background

OOTiA Handbook Rationale and DO-178B Compliance

» DO-178B was published in 1992 when procedural programming was the methodology de jour and does not specifically consider software developed using OOT

» OOT differs from the traditional functional approach to software development - satisfying some of the DO-178B objectives when using OOT may be unclear and/or complicated

» With no universal guidelines for using OOT in safety-critical systems, certification authorities have been using issue papers on a project-by-project basis to address OOT concerns

» The OOTiA Handbook extends the use of issue papers by identifying key issues and ***providing some guidelines*** to help the software community satisfy applicable DO-178B objectives when using OOT (Volume 3)

» It also provides ***an approach*** for certification authorities and their designees when evaluating OOT projects (Volume 4)

How to Use the OOTiA Handbook

OOTiA Handbook Background

OOTiA Handbook History and Future Plans

» The FAA and NASA co-sponsored the Object-Oriented Technology in Aviation (OOTiA) project to address OOT challenges in aviation

» Certification authorities, industry, and others submitted potential issues through a web site dedicated to the OOTiA project

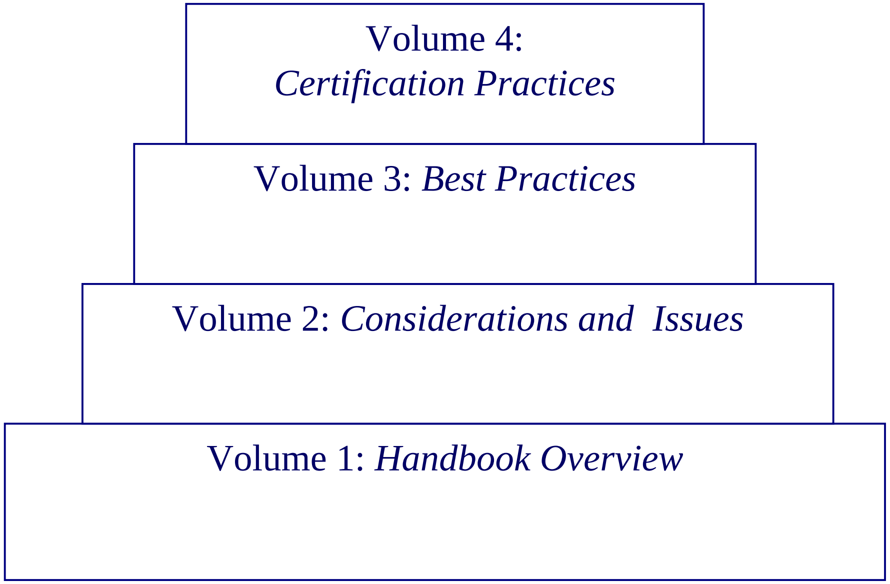
» OOTiA collaborators included the FAA, NASA, other government organizations, academia, international certification authorities, airborne systems manufacturers, and aircraft manufacturers

» The OOTiA Handbook and other documents may be used to impact future changes to the FAA’s software guidance (e.g., to impact future revisions to DO-178B or supplementary guidance)

» The OOTiA Handbook (or other guidance) will likely be updated in the future as OOT in aviation matures and lessons are learned

How to Use the OOTiA Handbook

OOTiA Handbook Overview


Volume 4: *Certification Practices*

Volume 3: *Best Practices*

Volume 2: *Considerations and Issues*

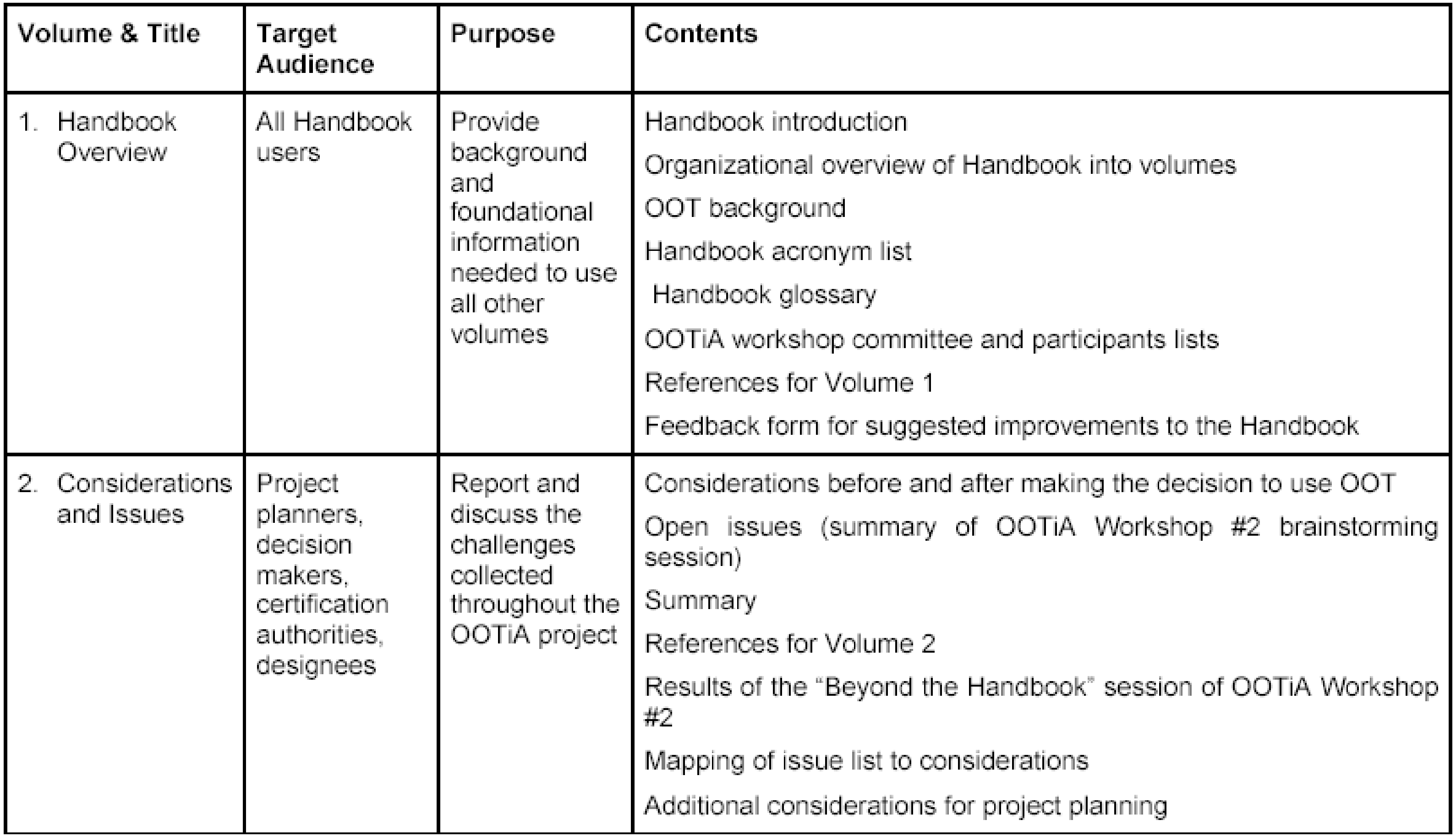
Volume 1: *Handbook Overview*

Note - The Handbook follows a “tiered” approach where each of the “lower” volumes provides the foundation for the “higher” volumes. For example, Volume 3: *Best Practices* relies on content in Volume 2: *Considerations and Issues* and Volume 1: *Handbook Overview* in substantiating its guidelines.

- Volume 1 provides the foundation which remaining Handbook sections build on as follows:
  - Handbook Introduction
  - Handbook Organization
  - OOT Background
  - Acronym List
  - Glossary
  - OOTiA Workshops
  - References
  - Feedback Form

How to Use the OOTiA Handbook

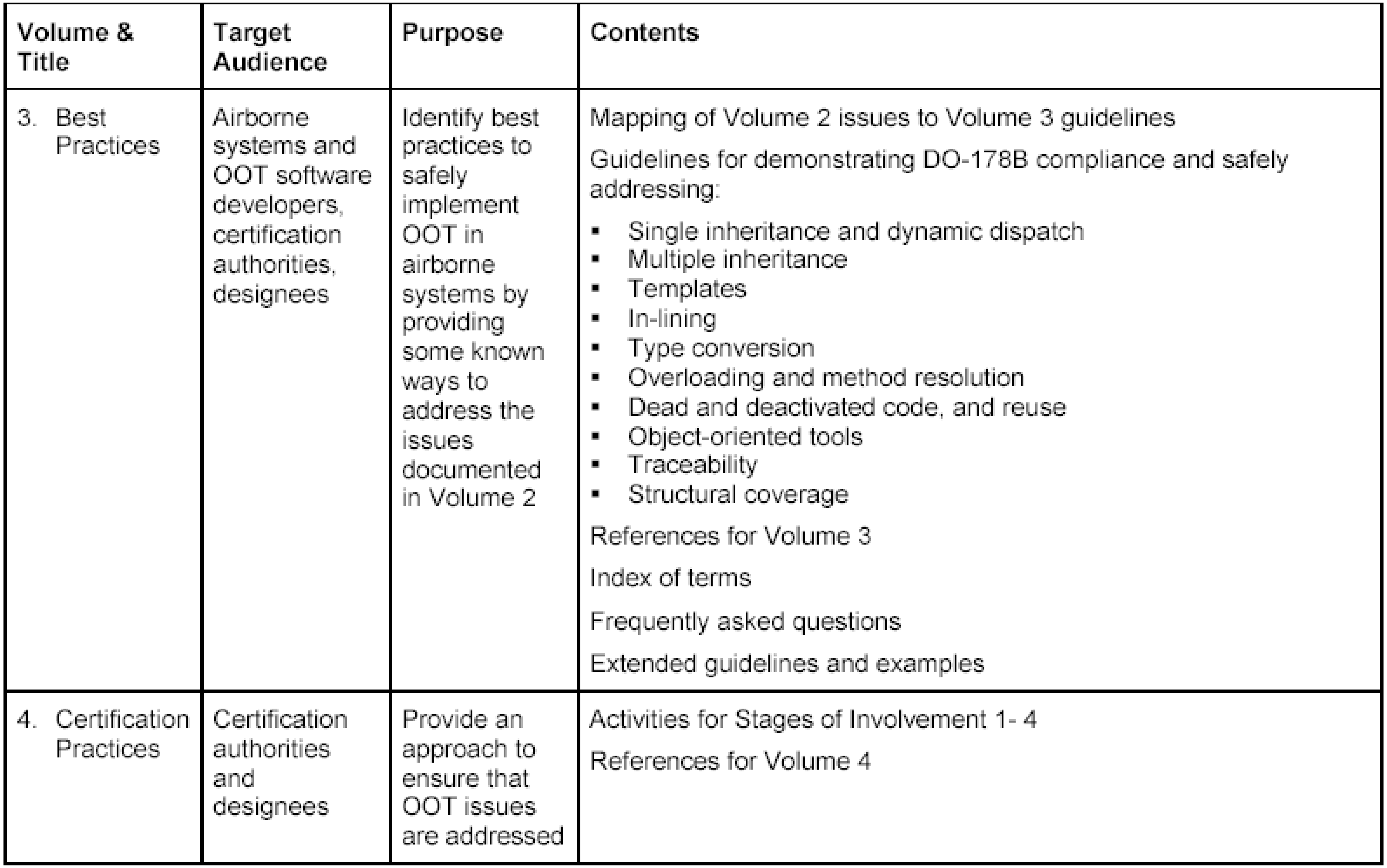
OOTiA Handbook Overview



Volume & Title	Target Audience	Purpose	Contents
1. Handbook Overview	All Handbook users	Provide background and foundational information needed to use all other volumes	Handbook introduction Organizational overview of Handbook into volumes OOT background Handbook acronym list Handbook glossary OOTiA workshop committee and participants lists References for Volume 1 Feedback form for suggested improvements to the Handbook
2. Considerations and Issues	Project planners, decision makers, certification authorities, designees	Report and discuss the challenges collected throughout the OOTiA project	Considerations before and after making the decision to use OOT Open issues (summary of OOTiA Workshop #2 brainstorming session) Summary References for Volume 2 Results of the "Beyond the Handbook" session of OOTiA Workshop #2 Mapping of issue list to considerations Additional considerations for project planning

How to Use the OOTiA Handbook

OOTiA Handbook Overview



Volume & Title	Target Audience	Purpose	Contents
3. Best Practices	Airborne systems and OOT software developers, certification authorities, designees	Identify best practices to safely implement OOT in airborne systems by providing some known ways to address the issues documented in Volume 2	Mapping of Volume 2 issues to Volume 3 guidelines Guidelines for demonstrating DO-178B compliance and safely addressing: ▪ Single inheritance and dynamic dispatch ▪ Multiple inheritance ▪ Templates ▪ In-lining ▪ Type conversion ▪ Overloading and method resolution ▪ Dead and deactivated code, and reuse ▪ Object-oriented tools ▪ Traceability ▪ Structural coverage References for Volume 3 Index of terms Frequently asked questions Extended guidelines and examples
4. Certification Practices	Certification authorities and designees	Provide an approach to ensure that OOT issues are addressed	Activities for Stages of Involvement 1- 4 References for Volume 4

How to Use the OOTiA Handbook

OOTiA Handbook Overview – Volume 1

- Handbook Introduction
  - Purpose
    - Identify key issues and provide ***potential*** approaches to address these issues when using OOT in airborne products
    - The Handbook provides ***an approach*** for certification authorities and their designees to help ensure that OOT issues have been addressed

From the OOTiA Handbook…

"NOTE:
This Handbook does not constitute Federal Aviation Administration (FAA) policy or guidance, nor is it intended to be an endorsement of object-oriented technology (OOT). This Handbook is not to be used as a standalone product but rather, as input when considering issues in a project-specific context."

How to Use the OOTiA Handbook

OOTiA Handbook Overview – Volume 1

- Handbook Organization
  - Scope
    - » The OOTiA Handbook is intended to be informational and educational
    - » Key OOT issues* which were identified as having ***potential impact*** in creating safe OOT-based airborne systems were addressed in the OOTiA Handbook (Note - Some identified issues are not unique to OOT but they are discussed because they may impact safe OOT implementations.)
    - » Some ***potential approaches**** were documented to address safety issues when using OOT in airborne systems
    - » Specific project criteria under which the OOTiA Handbook should or should not be used in software development is not defined – rather, determination is left to project planners, decision makers, or developers, as appropriate

* **Note – The OOTiA Handbook does not address all potential issues, nor are the guidelines in Volume 3 the only possible solutions to addressing the related issues.**

How to Use the OOTiA Handbook

OOTiA Handbook Overview – Volume 1

- OOT Background (very high-level summary)
  - OOT Basics
  - Principles of OOT
  - OOT Methodology
  - OOT Languages
  - Additional Key OO Concepts
    - Dynamic Dispatch
      - Single Dispatch
      - Multiple Dispatch
    - Liskov Substitution Principle (LSP)
  - Further OOT Reading
    - Reference resources for those who are interested in learning more about OOT

How to Use the OOTiA Handbook

OOTiA Handbook Overview – Volume 1

- Glossary
  - The glossary provides definitions for terms used in all volumes of the OOTiA Handbook and is intended to provide terminology for consistent communication and discussion of issues
  - Many terms are taken directly from the glossary of RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification,* others from the Object Management Group, while other terms are taken from references on object-oriented technology – references are noted, as appropriate, after each definition

- Rationale for OMG / UML Definitions
  - OMG is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. Membership includes virtually every large company in the computer industry, and hundreds of smaller ones
  - OMG members define and maintain the UML specification which are published in the series of documents for free download

Bottom line - the OMG is the closest thing to an “OOT standards body” for OOT at this point

How to Use the OOTiA Handbook

Where do I start? – Volume 2

If I were starting down the OO path and looking at a commercial, embedded avionics software product that must be approved, I would start with Volume 2 of the OOTiA Handbook

Volume 2 provides few answers – *potential* solutions are provided in Volume 3

How to Use the OOTiA Handbook

Overview – Volume 2

- There are two fundamental topics contained in Volume 2:
  - OOTiA Handbook historical data and intra-handbook links to ***some*** potential OOT approaches
  - A “devil’s advocate” view of OOT that could be used as an organizational self-check to determine if a project is ready to, or should, move into an OOT development effort

- Volume 2 raises awareness into areas such as the following:
  - Availability of resources (primarily trained staff and tooling)
  - Appropriateness (***or not***) of OOT
  - Considerations that need to be addressed in development, integration, and verification
  - ***Some*** potential regulatory issues (potential solutions not provided in Volume 2)
  - ***Some*** potential technical issues (potential solutions not provided in Volume 2)
  - Known open issues with OOT (pitfalls without clear answers)

- Given the above, the following stakeholders need to be cognizant of Volume 2
  - Project Planners
  - Decision Makers
  - Certification Authorities
  - Designees

How to Use the OOTiA Handbook

Tone – Volume 2

- Volume 2 tends to be quite negative in tone…
  - OOT compliance and safety issues raised during the OOTiA project were captured in Volume 2
  - Volume 2 tends toward “what can go wrong” vs. “what can go right”
  - The set of comments have been consolidated to provide a concise set of key concerns relevant to DO-178B compliance
  - The concerns documented in Volume 2 ***do not constitute*** a complete set of safety and certification concerns
  - Volume 2 does not discuss resolution of the concerns - Volume 3 does provide ***some*** guidelines for OOT software developers

How to Use the OOTiA Handbook

What do we do with Volume 2?

- Given that Volume 2 is a collection of issues and problems rather than answers and solutions, what do we do with it?
  - There are several possibilities depending on your perspective…
    - **Decision Makers**

      Volume 2 can be used to establish a set of questions, that when answered, will provide enlightenment about the readiness of the organization, including staff, tools and management, to determine what needs to be addressed before an OOT development effort is initiated
    - **Project Planners**

      Planners of projects can use Volume 2 early on to determine if the software planning process and documents (PSAC, SDP, SVP, CMP, SQAP, standards, et al) adequately cover concerns that will need to be addressed to develop safe and “certifiable*” OO software products
    - **Certification Authorities and Designees**

      Authorities can begin to use Volume 2 to enlighten themselves and determine if an applicant’s planning documents appear to address most of the concerns that may arise in an OOT software development effort (Note - Volume 4 contains a certification checklist/job aid approach)

* Generally software is not “certified” – systems are “certified” which contain “approved” software

How to Use the OOTiA Handbook

Appropriateness of OOT – Volume 2

- Considerations before the decision is made to use OOT

1. What are the benefits of OOT and are they quantifiable?
2. What project characteristics are important for or against OOT?
3. What project resources, specific to OOT are required?
4. What regulatory guidance exists for OOT programs and how will it be applied?
5. What are the technical challenges in applying OOT to ensure the appropriate level of integrity required for the project?

How to Use the OOTiA Handbook

Are OOT benefits quantifiable? – Volume 2

- What are the benefits of OOT and are they quantifiable?
  - OOT has become a popular software development approach due in part to claims related to reuse and efficient development
  - Studies exist that support the claimed benefits of modularity, reuse, encapsulation, and so on
  - Other studies show potential problems such as inferior understandability, complexity, and maintenance problems
  - There are no simple answers

Developers should carefully examine the evidence regarding OOT to better understand potential benefits and risks for their specific project.

How to Use the OOTiA Handbook

What are the project characteristics? – Volume 2

- What project characteristics are important for or against OOT?
  - Some of these attributes are
    - » **Size**
      - Larger programs ***may*** benefit from OO but, if this is a first attempt, smaller may be better
    - » **Criticality**
      - Higher criticality **may** indicate the organization should stay with what is familiar (this is part of what the Handbook tries to address)
    - » **Complexity** of the software
      - OO designs ***may*** allow better decomposition
    - » **Maturity** of the software requirements
      - Is this not true of either procedural or OO approaches?
      - Some argue OOT ***may*** be better for rapid iteration if requirements are immature
    - » **Applicability of OOT** to the specific problem domain
      - Standardized GUIs tend to be a classic application for OOT
    - » **Long-term plans for the product**
      - Is this a “throw-away” or “one-shot” effort or is it envisioned the product will be updated and maintained for many years?

Project resources – Volume 2

- What project resources specific to OOT are required?

  – **Personnel resources** – OOT specific training and experience including

  » OO methods for modeling, design, analysis and testing, and with OO tools

  » Experience of developers, testers and QA staff with OO techniques

  » Corporate resources available to understand and support OO techniques

  Note that training and experience are concerns for regulators also, including designees within the company and certification authorities responsible for the software approval on the project.

  – **Administrative resources** – Primarily standards

  » Industry standards for OOT – e.g., the OMG and UML

  » Standards for OO source code languages (e.g., Ada 95, Java, and C++)

  » Internal process standards (life cycle data standards and mapping to DO-178B)

  » Re-use standards for packaging of OO components

How to Use the OOTiA Handbook

OOT support tools – Volume 2

- What project resources specific to OOT are required?

  - **OO tools**
    - What is the compatibility of new OO tools with existing tools?
    - What type of integrated development environments (IDE’s) are available?
    - Is the organization familiar with the notations and processes the tools will require?
    - Are configuration management (CM) tools part of the IDE and, if not, how will CM be handled (especially for between models and source code)?
    - How will traceability and change impact analyses be handled?
    - Will tools need to be qualified, and if so, what will it cost?
    - Tools may introduce levels of abstraction that do not clearly map to DO-178B – has it been determined how this will be handled?

How to Use the OOTiA Handbook

Why do project resources matter? – Volume 2

From the OOTiA Handbook…

“The project characteristics together with the OOT specific resources within a company will influence the level of involvement, or degree of oversight, that the FAA has with a project. This is a non-trivial consideration with respect to both time and cost. The level of FAA involvement will dictate the number of software reviews, the stages of involvement, and the nature of the review.”

How to Use the OOTiA Handbook

Regulatory guidance – Volume 2

- What regulatory guidance exists for OOT and how will it be applied?
  - DO-178B must still be satisfied
  - The OOTiA Handbook provide “guidelines” or “best practices”* on how to use OOT and still comply with DO-178B
  - The OOTiA Handbook is intended to provide clarification until further supplemental guidance is established (perhaps SC-205)

The bottom line is that certification authorities, applicants, system developers, and software developers should understand the requirements that an OO system must satisfy for it to be approved, and how the system will be shown to satisfy these requirements. The earlier this is understood and documented in appropriate issue papers, planning documents and so on, the lower the risk will be to the development effort.

*** From the OOTiA Handbook…**

“…the Handbook is not intended to become official FAA policy or guidance; it will likely influence the approval process for an OO program, and will likely influence FAA’s future software guidance.”

How to Use the OOTiA Handbook

Requirements challenge – Volume 2

- Technical challenge of requirements
  - There are two fundamental concerns with respect to OO approaches for requirements elucidation as compared to procedural or functional approaches
    - » There are differences between approaches to requirements decomposition (functional versus object)
    - » The documentation of requirements may be different between procedural and OO approaches

How to Use the OOTiA Handbook

Requirements challenge – Volume 2

- Technical challenge of requirements (decomposition)
  - Does the distribution of functionality inherent in OO systems complicate assurance of the intended functionality?
    - » OO program flow may involve calls between many small methods defined by different classes and, thus, complicate functionality implementation
    - » On the other hand, distribution of functionality may include the means to encapsulate state, eliminate references to global data, and more easily ensure synchronization of processes

How to Use the OOTiA Handbook

Requirements challenge – Volume 2

- Technical challenge of requirements (documentation)
  - Both approaches typically have software requirements data comprised of text with diagrams maintained by modeling tools
  - Functional/procedural diagrams tend to consist largely of data and control flow diagrams, structure charts, entity-relationship charts, state and sequence diagrams, data stores, and data schemas
  - OOT diagrams typically consist of use cases, associations, packages, classes, subtypes, objects, and activity diagrams
  - OOT approaches can make it difficult to determine how to map the OOT diagrams and their subsequent refinements to the DO-178B objectives and life cycle data

- Thought question (freebie):
  - **How do “traditional” data flow diagrams, control flow diagrams, state diagrams or entity relationship diagrams fit into normative mapping of DO-178B for life cycle data?**

Before the OOT Decision – Technical Challenges

From the OOTiA Handbook…

Requirements definition by any method is a significant challenge to developing a correct and safe system. While some would argue that the close correspondence between real-world entities and the objects within an OO requirements model makes it easier to construct, validate, and maintain such models, developers should consider whether OOT makes this challenge easier or more difficult for their project.

Verification challenge – Volume 2

“When abstraction works well, it hides details that do not need to be tested, so testing becomes easier*. When abstraction works poorly, it hides details that need to be tested, so testing becomes harder (or at worst, infeasible). Typically, testing must occur at some compromise level of abstraction, gaining leverage from the abstractions that clarify the software's functionality while penetrating the abstractions that obscure it. Often, a software developer's level of familiarity with object-oriented development techniques is a key factor in determining whether abstraction is used in a positive or negative way, which in turn determines whether testing is helped or hindered by the unique features of object-oriented languages.”

NIST Special Publication 500-235, Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, Arthur H. Watson & Thomas J. McCabe

“Object-oriented programs are generally more complex than their procedural counterparts. This added complexity results from inheritance, polymorphism, and the complex data interactions tied to their use. Although these features provide power and flexibility, they increase complexity and require more testing.”

Roger T. Alexander, http://www.cs.colostate.edu/~rta/publications/s5qua01.pdf

* - Not necessarily a DO-178B perspective, but illustrative in any case

How to Use the OOTiA Handbook

Verification challenge – Volume 2

- Technical challenge of verification
  - There are several OO verification issues of concern (many are related to the notion of inheritance)
    - » Source code to object code traceability issues
    - » Low-level requirements-based test coverage may be a challenge (What are low level requirements for OOT??
    - » Robustness test coverage may be more difficult
    - » Control flow and data flow analyses may be very difficult or perhaps infeasible depending on how the OO implementation was handled

From the OOTiA Handbook…

Do we have the same level of confidence in our ability to adequately review, analyze, and test OO programs as we do with traditional functional programs?

How to Use the OOTiA Handbook

Verification challenge – Volume 2

- Technical challenge of verification
  - Are additional verification methods needed for OOT to achieve the same level of assurance that could be obtained under a functional approach?

More research is needed to fully understand error classes that are unique to OOT and to better understand the extent to which existing methods are adequate for verifying OOT.

How to Use the OOTiA Handbook

Safety challenge – Volume 2

- Technical challenge of safety
  - There is considerable debate about system safety assessments with respect to OOT development efforts
  - Current safety analysis is often based on determining that a function, with a high degree of cohesion and low coupling, as implemented, is correct and safe
  - Conversely, OO-based operations related to a function, ***can*** be widely distributed among objects that interact with each other by exchanging messages - assessing these interactions ***may*** complicate the safety analysis and verification activities

From OOTiA Handbook…

Although the system safety assessment is not one of the software life cycle processes specified in DO-178B, connections with system development and system safety assessment are mentioned in DO-178B. Hence, the effect of OO design and implementation on the safety assessment process should be considered and addressed.

So your organization has decided to take the OO plunge…

How to Use the OOTiA Handbook

Key concerns – Volume 2

- There are a series of topics, with a number of subtopics, to be addressed after the organization has decided to wade into the OO waters…
  - Planning
  - Requirements, Design, Code, Integration
  - Integral Processes
  - Additional Considerations

Note - This section will establish the rationale for Volume 3.

How to Use the OOTiA Handbook

Planning considerations – Volume 2

- Planning
  - One of the biggest planning considerations is the manner in which OO processes and life cycle data will map into DO-178B
  - Concerns include the following
    - » Life cycle data definitions
    - » Definition of requirements methods
    - » Definition of requirement notations
    - » Restriction of potentially unsafe or non-verifiable techniques (e.g., multiple inheritance)

How to Use the OOTiA Handbook

Planning considerations – Volume 2

- Defining Life Cycle Data
  - OOT introduces new notations, models and terminology such as use cases and various diagrams* (class, sequence, component, deployment, activity and state chart diagrams)
  - Given these various schemas, which will likely be combined, how will the organization map to the DO-178B section 11 life cycle data – e.g., what is “code”, “low-level requirements”, “high-level requirements” or “architecture”?
  - In a traditional DO-178B program, software requirements standards, as described in section 11.6 of DO-178B, define the methods, rules, notations, and tools for developing high-level requirements – how will this be handled for an OO effort which may be much “richer” in terms of “methods, rules, notations and tools”?

* **Note, there are also functional/procedural types of diagrams that do not map directly to DO-178B life cycle data descriptions either.**

How to Use the OOTiA Handbook

Planning considerations – Volume 2

- Requirements Methods And Notations
  - DO-178B compliance focuses on a software functional perspective starting with top-down software requirements, design and verification, including requirements and structural coverage analyses
  - Graphical OOT methods and notations can be a concern* in terms of full traceability to lower level requirements and source code as well as identification of derived requirements
  - OOT development focuses on "objects" and the "operations" performed by or on those objects and may actually be a combination of top-down, bottom-up and re-use techniques that may not ***easily*** provide equivalent levels of design assurance
  - Assuming there is re-use occurring on OO developments, derived requirements and their link to the safety assessment may require particular attention
  - If formal methods are to be used, additional techniques may need to be discussed in the planning documents for both development and verification activities (Note, this also applies to functional/procedural development efforts.)

* **OO notations still allow for rich textual or algorithmic specifications but the approach is often not emphasized in many OO language classes provided by universities. As one gets deeper into OO analysis, it becomes obvious that detailed specifications are still required.**

How to Use the OOTiA Handbook

Restrictions in standards – Volume 2

- Restrictions
  - Section 4.5c of DO-178B states, “the software development standards should disallow the use of constructs or methods that produce outputs that cannot be verified or that are not compatible with safety-related requirements”
  - Some OO languages have features that could make it extremely difficult or impossible to satisfy the objectives of DO-178B* - an example of this could be the indiscriminate use of single or multiple inheritance

* **There are also features of some procedural languages that would make it extremely difficult or impossible to satisfy the objectives of DO-178B (several examples would include unlimited recursive routines or pointer issues leading to memory leaks).**

From OOTiA Handbook…

“The key concern is that language features, such as multiple inheritance, should be evaluated carefully in the planning process and restrictions or rules established, documented, and followed as warranted for a particular project.”

How to Use the OOTiA Handbook

Requirements, design, code and integration – Volume 2

- Requirements, design, code and integration
  - The vast majority of current concerns with OOT efforts is in the area of requirements, design, code and integration – within those major areas there are a number of sub-areas we need to be concerned with
    - Subtypes
      - Type Substitutability
      - Inconsistent Type Usage
    - Subclasses
      - Unclear Intent
      - Overriding
    - Memory Management and Initialization
      - Indeterminate Execution Profiles
      - Initialization
    - Dead or Deactivated Code
      - Identifying Dead and Deactivated Code
      - Libraries and Frameworks

How to Use the OOTiA Handbook

Memory management – Volume 2

- Memory Management and Initialization

  - Memory management

    - Per DO-178B (section 11.10), developers need to be concerned with memory management and the memory management strategies and limitations – there are some aspects of OO techniques that can make memory management a little more challenging

    - Typically, most real-time, safety critical, embedded systems have avoided the use of dynamic memory allocation and de-allocation – this should still hold for OO, but naïve approaches to object creation can lead to memory management issues (e.g., creating objects “on-the-fly” with no limit on the number of objects that may be created and inappropriate destructors)

    - For either procedural or OO approaches, dynamic memory allocation and de-allocation is a risky proposition that bears intense scrutiny in terms of timing and deterministic behavior

How to Use the OOTiA Handbook

Memory management – Volume 2

- Memory Initialization
  - Depending upon the compiler and linker, class hierarchies (deep hierarchies in particular) may lead to object initialization problems
  - An object’s constructor method should always be invoked before any other method within the object is invoked
  - Depending on the compiler, it is possible for some other operation than the constructor to be executed first
  - If the constructor is not executed properly, the object or object state variable(s) could be initialized to incorrect state(s)

How to Use the OOTiA Handbook

Dead or deactivated code – Volume 2

- Dead or Deactivated Code
  - Reuse is an important design goal for many programs - either OO or procedural
  - But there is a catch - requirements, design, and code of a reusable component might contain more functionality than required by the system being certified raising concerns about dead and deactivated code
  - Since it is believed there may be more re-use with OOT approaches, the dead and deactivated code issue may become larger in newer developments

How to Use the OOTiA Handbook

Dead or deactivated code – Volume 2

- Identifying dead or deactivated code
  - DO-178B (section 6.4.4.2) requires that dead code be removed and analysis performed to assess the effect and need for re-verification
  - Similarly, DO-178B (section 5.4.3) requires deactivated code to be verified (analysis and test) to prove that it cannot be inadvertently activated
  - Dead and/or deactivated code can be difficult to identify in OO applications – consider the following:
    - » Not all methods of a class are called in a particular application (think of libraries as an example)
    - » Methods of a class are overridden in all subclasses (think of abstract classes)
    - » Attributes (data) of a class are not accessed in a particular application

How to Use the OOTiA Handbook

Libraries – Volume 2

- Libraries and Frameworks
  - Libraries and frameworks may present challenges to DO-178B compliant developers
    - » COTS libraries are often not developed with safety-critical applications in mind
    - » Developers will often use only a portion of the library or framework functionality - dead and/or deactivated code can result

Libraries and frameworks ***may*** be more pervasive in an OO effort - the use of libraries must be carefully considered, addressed and verified* for proper functionality

* Recall verification requires requirements data, design data, source code, requirements coverage analysis, and structural coverage analysis – this could be very challenging for third party, commercial software

How to Use the OOTiA Handbook

Integral processes – Volume 2

- ***Integral processes*** cover a wide slice of DO-178B – including verification
  - Verification concerns include
    - Flow analysis
    - Structural coverage analysis
    - Timing and stack analysis
    - Source to object code traceability
    - Requirements based testing
    - Test case re-use
  - Configuration Management (CM) (concerns previously discussed)
    - Configuration identification and control
  - Traceability (concerns previously discussed)
    - Function vs. object tracing and complexity
    - Tracing through OO views
    - Iterative development

How to Use the OOTiA Handbook

Verification – Volume 2

- Verification
  - Simply put, verification examines the relationship between the software implementation and the requirements to detect and report errors introduced during the development processes
  - At this point in the history of DO-178B, error mechanisms for functional/procedural development efforts are fairly well understood – conversely, OO techniques, especially dynamic dispatch and polymorphism, can challenge and complicate various traditional verification activities

We will quickly hit a number of implementation issues leading to verification concerns - the key in all of these concerns is to understand exactly what the compiler is doing when various language features are implemented (and create/enforce coding restrictions, as appropriate)

How to Use the OOTiA Handbook

Verification cont'd – Volume 2

- Flow Analysis - Polymorphism and dynamic dispatch
  - Data flow and control flow analyses must be performed for software levels A, B, and C to confirm correct data and control coupling between code components; however…
  - Program control flow (execution) can be difficult to predict in OO implementations because dynamic dispatch/polymorphism forces late method binding
  - Because of late binding, a single line of source code ***may*** mean many different things depending upon specific data values that the program encounters during execution (Recall our earlier example of a "super-class list" that could contain subclass objects.)
  - Determining the correctness of control flow decisions requires analysis of how individual objects control the execution flow of the software – control flow decision points ***may*** use data objects with values that are maintained in other parts of the software making flow analysis difficult

How to Use the OOTiA Handbook

Verification cont'd – Volume 2

- Flow Analysis – Abstraction and in-lining
  - Abstraction allows the attributes (data) associated with an object to be private - access is only provided through the object's methods
  - This can complicate the analyses in terms of gaining direct access of the attributes – on the other hand, limiting access can also simplify analysis because it is clear that the attributes may only be changed in one place with the rules defined by the method that operates on the attribute(s)
  - In-lining can also complicate flow analysis (not just an OO concern) as it can cause substantial differences between the apparent flow in the source code and the actual flow in the object code

How to Use the OOTiA Handbook

Summary – Volume 2

- **The Volume 2 summary covers two pages –**
  - **It is a list of issues would be *a good place to start* once a commitment has been made to investigate and potentially pursue an OOT development effort**

Concentrate on Volume 2 – it will provide a list of questions you need to be able to answer if you are going down the OO path

How to Use the OOTiA Handbook

Introduction – Volume 3

- The primary purpose of Volume 3 is to cite a number of “best practices*” to deal with OOT concerns including the following:
  - Single Inheritance and Dynamic Dispatch
  - Multiple Inheritance
  - Templates
  - Inlining
  - Type Conversion
  - Overloading and Method Resolution
  - Dead And Deactivated Code, And Reuse
  - Object-Oriented Tools
  - Traceability
  - Structural Coverage

* **The best practices are called “*__guidelines__*” and may have associated “*__rules__*” in the handbook. The *__rules and guidelines are not regulations__*, but if not adhered to, the applicant will need to provide effective, alternative solutions to address these issues.**

How to Use the OOTiA Handbook

Background – Volume 3

- **Volume 3 was written as a snapshot in time…**

From the OOTiA Handbook…

OOT in embedded and safety-critical systems is still an evolving discipline; best practices will likely be added and modified in … future revisions of this Handbook. In any case, the OOT standards and methods the developer intends to use should be documented in the project plans and standards, and presented to the certification authorities as early as possible in the project.

How to Use the OOTiA Handbook

Intended audience – Volume 3

- Given the implementation and hands-on nature of Volume 3, there are several target audiences
  - Developers
    - Those who will create (implement requirements into source code), verify and maintain the software
  - Certification Authorities
    - Those who will ultimately approve the systems containing the software which is being developed
  - Designees
    - Those who, depending on their level of delegation, may approve or assist in approving software contained in systems to be certified as well as providing knowledgeable day-to-day guidance to their employers

How to Use the OOTiA Handbook

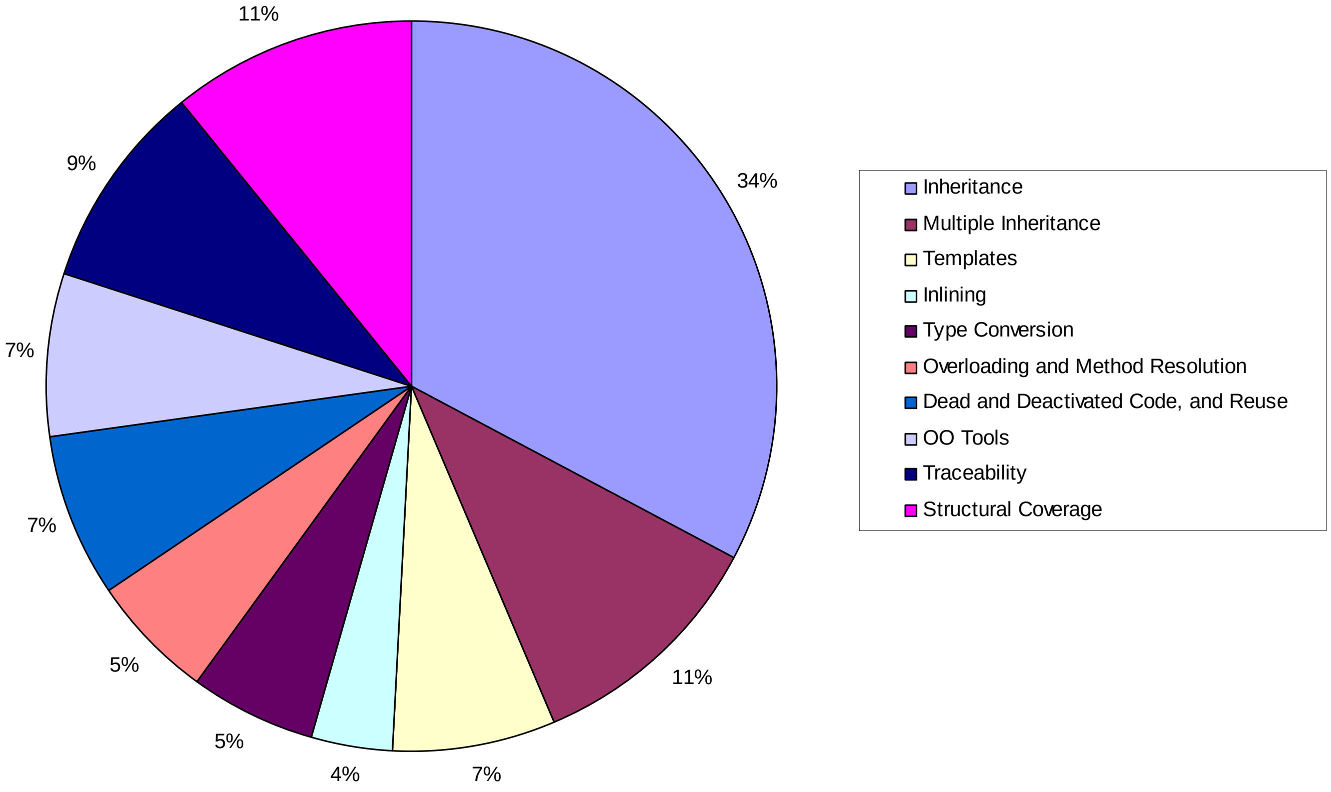
Overview and layout – Volume 3

3.1	INTRODUCTION	
3.2	MAPPING OF VOLUME 2 ISSUES TO VOLUME 3 GUIDELINES	
3.3	*SINGLE INHERITANCE AND DYNAMIC DISPATCH*	
3.4	MULTIPLE INHERITANCE	
3.5	TEMPLATES	
3.6	INLINING	
3.7	TYPE CONVERSION	
3.8	*OVERLOADING AND METHOD RESOLUTION*	
3.9	DEAD AND DEACTIVATED CODE, AND REUSE	
3.10	OBJECT-ORIENTED TOOLS	
3.11	*TRACEABILITY*	
3.12	*STRUCTURAL COVERAGE*	
3.13	REFERENCES	
3.14	INDEX OF TERMS	
	APPENDIX A	FREQUENTLY ASKED QUESTIONS (FAQS)
	APPENDIX B	EXTENDED GUIDELINES AND EXAMPLES

How to Use the OOTiA Handbook

Overview and layout – Volume 3

- Another way to gain appreciation for the topics emphasized in Volume 3 is to simply examine the number of pages dedicated to each topic…



How to Use the OOTiA Handbook

How do I use it? – Volume 3

- As has been stated, Volume 3 is a collection of “best practices” that are called “guidelines” or “rules”, which should be employed if OO techniques are utilized
  - These “best practices” or “guidelines” are intended to address ***key*** concerns and issues raised in Volume 2
  - The “best practices” or “guidelines” are ***not the only way*** to address issues raised in Volume 2 – there may be other alternatives
  - ***In some cases, “guidelines” are preliminary or incomplete*** (e.g., structural coverage analyses and data coupling and control coupling) – in these cases, the applicant must define, document and seek approval for their particular means of compliance

How to Use the OOTiA Handbook

How do I use it? – Volume 3

- Some Volume 3 quotes worth remembering…

From the OOTiA Handbook…

OOT in embedded and safety-critical systems is still an evolving discipline; best practices will likely be added and modified in the future revisions of this Handbook.

In all cases, it should be noted that it is still the developer's responsibility to demonstrate that the OOT methods and processes they have selected to utilize can, and will, provide the appropriate integrity for safe software implementation.

In any case, the OOT standards and methods the developer intends to use should be documented in the project plans and standards, and presented to the certification authorities as early as possible in the project.

How to Use the OOTiA Handbook

Potential answers – Volume 3

- ***Key Concerns/Issues Addressed by the Guidelines***

  - Volume 3 provides a reference between Volume 2 and 3 in Table 3.2-1 of the Handbook (excerpt on the following slide)

  - Volume 3 ***may*** provide multiple means to address a key concern - in these cases, only <u>one</u> guideline of those listed (separated by OR) needs to be utilized

  - Not all Volume 2 issues were addressed – this is because of one or more of the following reasons:
    - » The issue is not specific to OOT (although we will look at issues that are not OO-unique)
    - » The topic is an area of active research where no guidelines currently exist
    - » The issue is a concern not categorized in Volume 2

How to Use the OOTiA Handbook

Mapping to Volume 2 – Volume 3

- Sample extract of linkages between Volumes 2 and 3

Key Concern (from Volume 2)	**Issue List #**	**Guidelines (in Volume 3)**
Volume 2, section 2.3.1.1 How does the life cycle data from an OO development process map to the life cycle data specified in DO-178B?	**77, 87**	Sections 3.10.3, 3.11.4.1, and 3.11.6.1
Volume 2, section 2.3.1.2 Are OO approaches adequate to define all types of requirements? Specifically, can we capture all nonfunctional requirements of interest? And can we avoid problems associated with graphical grouping?	**63, 75, 78, 79, 80**	Sections 3.10.3, 3.11.5.1, 3.11.6.1, and 3.11.9 IL 78 and IL 80 may not be completely addressed in Volume 3.
Volume 2, section 2.3.1.2 A well-defined means to map formal specifications to natural language and/or other less formal notations (e.g. UML) is needed to make formal specifications generally accessible.	**73**	Not addressed in Volume 3.

How to Use the OOTiA Handbook

How will my software be evaluated? – Volume 4

- Very straightforward layout…
  - Activities for Stages of Involvement
    - *Activities for Stage of Involvement #1 (SOI #1) – Planning Review*
    - *Activities for Stage of Involvement #2 (SOI #2) – Development Review*
    - *Activities for Stage of Involvement #3 (SOI #3) – Verification Test Review*
    - *Activities for Stage of Involvement #4 (SOI #4) – Final Review*

How to Use the OOTiA Handbook

Linkage to the SOI Job Aid – Volume 4

- The SOI software reviews (Job Aid) are the primary means for certification authorities and designees to assess compliance to DO-178B objectives

- Volume 4 provides a series of activities and questions to be considered ***in conjunction with those in the Job Aid***

- Each SOI activity has a set of questions designed to help carry out that activity much like the Job Aid

  – The questions are not all encompassing

    » **Some questions will not be applicable to all OOT projects**

    » **Some projects will require additional questions**

  – Each question includes reference(s) to DO-178B and Volume 2 of this Handbook.

How to Use the OOTiA Handbook

Target audience – Volume 4

From the OOTiA Handbook…

This volume provides an approach for certification authorities and designees to ensure that OOT issues have been addressed in the projects they are reviewing or approving.

How to Use the OOTiA Handbook

Supplemental job aid questions – Volume 4

- The following slides are a straight cut n’ paste from the Handbook (one exception on the question pertaining to initialization)
- Some of the more challenging OO issues have been offset in red, bold font (will simply be bold font if you do not have color slides)
- The supplemental questions are not perfect – please make a note how they can be improved if you feel improvement is in order

How to Use the OOTiA Handbook

SOI 1 – Planning

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
1.1	**Review all software plans and consider the following questions:**		
1.1.1	**Does the additional considerations section (or some other section) of the Plan for Software Aspects of Certification (PSAC) address that the project is object-oriented (OO) and identify OO-related issues for the specific project?**	2.3.4	A1: 4
1.1.2	Have the issues raised in Volume 2 of the Handbook (summarized in section 2.5.2 of Volume 2) been addressed by the applicant's plans? Note: Volume 3 provides some approaches for addressing OO issues; however, **applicants may have other means to address the issues.**	2.3.1, 2.5.2	All
1.1.3	Do the plans address how the following common OO implementation features will be addressed to comply with DO-178B objectives? 1. Encapsulation 2. Overloading **3. Inheritance** **4. Dynamic binding/dispatch** **5. Polymorphism** 6. Templates 7. Inlining 8. Deactivated code 9. Reusability Note: Applicants may not use all of the features available.	2.3.1	A1: 4

How to Use the OOTiA Handbook

SOI 1 – Planning

Item #	**Evaluation Activity/Question**	**OOTiA Vol 2 Ref**	**DO-178B Obj**
1.1.4	**Do the plans define how the OO life cycle data maps to the DO-178B Section 11 life cycle data? For example, what are the high-level requirements, low-requirements, design, and source code in OO?**	2.3.1.1	A1: 1-4 (Sect 11.1e)
1.1.5	What levels of abstraction are used for the OO implementation and how does the abstraction map to the DO-178B objectives (reference 2.3.1.1 of Volume 2)?	2.3.1.1	A1: 1
1.1.6	Do plans identify how derived requirements will be addressed and tracked in the OO implementation?	2.3.1.2	A1: 1
1.1.7	Has the difference between dead and deactivated code for OO been clarified in the planning phases of the program?	2.3.2.4	A1: 4
1.1.8	**If a modeling language is used (e.g., UML), how is the software functionality described and how is the tie to the system safety assessment established?**	2.3.1.2	A1: 1

How to Use the OOTiA Handbook

SOI 1 – Planning

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
1.2	**Review the Development Standards (requirements, design, & coding standards) and consider the following questions:**		
1.2.1	**Do the development standards address limitations for OO implementation** (e.g., identify project rules that restrict use of specific target language features)?	2.3.1.1 and 2.3.1.2	A1: 5 (Sect 11.6-11.7)
1.2.2	Are OO methods and notations documented in the software requirements and design standards? Specifically, **do the standards identify how non-functional requirements are captured, and how low-level and derived requirements are addressed?**	2.3.1.1 and 2.3.1.2	A1: 5 (Sect 11.6-11.7)
1.2.3	Do the development standards implement the solutions to the OO issues as the PSAC (or other plans) claim they will?	N/A	A1: 5 & 7
1.2.4	**Do the standards disallow "constructs or methods that produce outputs that cannot be verified or that are not compatible with safety-related requirements"** (per DO-178B, section 4.5c)?	2.3.1.3	A1: 5 (Sect 4.5c)
1.2.5	Are necessary restrictions for the chosen OO language identified in the coding standards?	2.3.1.3	A1: 5 (Sect 11.8)
1.2.6	**If constructs or methods were disallowed, do the plans address how proper implementation will be verified?**	2.3.1.3	A1: 1,2,5,&7

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.1	**Assess the developer's traceability approach in the OO implementation by considering the following questions:**		
2.1.1	How is traceability between requirements, design, code, and test cases/procedures established and maintained?	2.3.3.4	A3:6 A4:6 A5:5
2.1.2	**Does the traceability approach address common OO traceability problems**, such as: • traceability of functional requirements through implementation might be lost or difficult with an OO program because of mismatches between function-oriented requirements and an object-oriented implementation • complexity of class hierarchies may cause traceability problems • the behavior of the classes and how they interact to provide the required function might not be visible in any single view.	2.3.3.4	A3:6 A4:6 A5:5
2.1.3	Does the traceability approach provide a way to ensure that: • all requirements (both high and low-level) are verified, • unintended functions and/or source code are identified, and • visibility into derived requirements exists?	2.3.3.4	A3:6 A4:6 A5:5
2.1.4	**Do the OO tools/methods support traceability across the full life cycle and across multiple views?**	2.3.3.4	A3:6 A4:6 A5:5

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.2	**Determine if subtype issues have been addressed by considering the following questions:**		
2.2.1	**How has the applicant addressed type substitutability?** For example, has the applicant provided assurance that subtype substitution is always appropriate? If the Liskov Substitution Principle (LSP) has been adopted, have language issues been clarified?	2.3.2.1	A2, A3, & A4
2.2.2	**Have inconsistent type uses been addressed?**	2.3.2.1.2	A2, A3, & A4
2.2.3	**If a language subset is proposed, does it address the subtype issues?**	2.3.2.1	A2, A3, & A4

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.3	**Determine if subclass issues have been addressed by considering the following questions:**		
2.3.1	**Is the intent of subclass operations clear and unambiguous?** Specifically, consider: • Is the complete definition/intention of a class clear (in order to avoid a subclass being incorrectly located in a hierarchy)? • How is the class hierarchy accuracy assured? • Do top-heaviness or deep hierarchies exist (six or more subclasses)? If so, how is it verified that the wrong variable type, variable, or method are not inherited? • How are unintended connections among classes (which could lead to difficulty in meeting the DO-178B/ED-12B objective of data and control coupling) prohibited?	2.3.2.2.1	A3: 1-4, 6,7 A4: 1-4, 6-11
2.3.2	Have overriding concerns been addressed? Specifically: • Can an operation accidentally override another? • Can operations inherited from different sources be accidentally joined? • Can a subclass-specific implementation of a superclass be accidentally omitted? • Have the classes of errors associated with overriding that were identified in Volume 2 (section 2.3.2.2.2.d) been addressed?	2.3.2.2.2	A3: 1-4, 6,7 A4: 1-4, 6-11, 13

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.4	**If objects or design components are being reused, have dead/deactivated code issues been addressed? Specifically:**		
2.4.1	Are the issues of dead and deactivated code addressed consistently throughout the project and as the planning documents stated?	2.3.2.4.1	A5 & A7 (Sect 5.4.3 & 6.4.4.3)
2.4.2	Does overriding result in any dead or deactivated code?	2.3.2.4.1	A5 & A7 (Sect 5.4.3 & 6.4.4.3)
2.4.3	Do libraries and OO frameworks result in dead or deactivated code?	2.3.2.4.2	A5 & A7 (Sect 5.4.3 & 6.4.4.3)

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.5	**Does the OO approach address the following configuration management issues:**		
2.5.1	Is configuration management maintained when objects and classes are used multiple times in slightly different manners?	2.3.3.3	A8
2.5.2	**If a modeling tool is used, how is the configuration of objects and classes handled?**	2.3.3.3	A8
2.5.3	**How is configuration control maintained for inherited classes, association links, and client relationships?**	2.3.3.3	A8

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.6	**If tools are used in the OO implementation or verification, consider the following questions:**		
2.6.1	Does the tool support compliance with the DO-178B objectives? i.e., is the tool compatible with DO-178B?	2.3.4.1	Objs vary depending on the tool (Sect 12.2)
2.6.2	How mature is the tool (e.g., compiler, UML tool, …) that is being used? Unstable configuration may indicate an immature tool.	2.3.4.2	Objs vary depending on the tool (Sect 12.2)
2.6.3	**How will the tool be maintained to meet the long-term needs of an aircraft project?**	2.3.4.2	A8: 6 (Sect 7.2.9)

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.6.4	**If a tool does not meet the DO-178B definition of development or verification tool, how is it addressed?** i.e., are new classes of tools being used? If so, is an issue paper needed? Does the tool need to be qualified?	2.3.4.3	Objs vary depending on the tool (Sect 12.2)
2.6.5	If the tool is being qualified, does it meet DO-178B and applicable guidelines in Order 8110.49?	2.3.4.3	Objs vary depending on the tool (Sect 12.2)

How to Use the OOTiA Handbook

SOI 2 – Development

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
2.7	**Consider if the following potential issues have been addressed:**		
2.7.1	Is garbage collection used? If so, **how is determinism established?**	2.4	A2, A3, A4, A5
2.7.2	**Is exception handling used?** If so, how are control flow, run-time support, determinism, and deactivated code addressed?	2.4	A2, A3, A4, A5
2.7.3	**How is concurrency addressed?** Use of this feature may raise issues with control flow, run-time support, real-time predictability, and deactivated code.	2.4	A2, A3, A4, A5
2.7.4	How is any additional complexity managed in OO projects?	2.4	All
2.7.5	**If libraries are used, do they meet DO-178B objectives?**	2.3.2.4.2	All

How to Use the OOTiA Handbook

SOI 3 – Verification and Test

Item #	**Evaluation Activity/Question**	**OOTiA Vol 2 Ref**	**DO-178B Obj**
3.1	**Consider if data and control coupling issues have been addressed by considering the following questions:**		
3.1.1	**Does the OO approach address data and control coupling analysis?**	2.3.3.1.1	A7: 8
3.1.2	**If dynamic dispatch is used, how is data and control coupling analysis performed?**	2.3.3.1.1	A7: 8
3.1.3	If inlining is used, how is data and control coupling analysis performed?	2.3.3.1.1	A7: 8

How to Use the OOTiA Handbook

SOI 3 – Verification and Test

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
3.2	**Determine if memory management and initialization issues have been addressed by considering the following questions:**		
3.2.1	**Does the memory allocation/deallocation approach result in predictable worst-case memory analysis?**	2.3.2.3	A6 (Sect 6.4.3a)
3.2.2	**Is memory fragmentation and defragmentation handled in a deterministic manner?**	2.3.2.3.1	A6 (Sect 6.4.3a)
3.2.3	**Have memory leaks been addressed?**	2.3.2.3	A6 (Sect 6.4.3a)
3.2.4	Have initialization problems been identified and addressed? (e.g., constructors, shallow vs. deep copy, etc)	2.3.2.3.2	A6 (Sect 6.4.3a)

How to Use the OOTiA Handbook

SOI 3 – Verification and Test

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
3.3	**Consider if structural coverage issues have been addressed by considering the following questions:**		
3.3.1	**If dynamic dispatch is used, how is structural coverage addressed?**	2.3.3.1.2	A7: 5-7
3.3.2	**If inheritance is used**, how is structural coverage addressed?	2.3.3.1.2	A7: 5-7
3.3.3	**If polymorphism is used**, how is structural coverage addressed?	2.3.3.1.2	A7: 5-7
3.3.4	If inlining is used, how is structural coverage addressed?	2.3.3.1.2	A7: 5-7
3.3.5	If templates are used, how is structural coverage addressed?	2.3.3.1.2	A7: 5-7
3.3.6	**If source to object code traceability is required (per DO-178B, section 6.4.4.2b), how is it carried out?** Specifically, how do dynamic dispatch, inlining, and type conversion affect source to object code traceability?	2.3.3.1.4	A7: 7
3.4	**If dynamic dispatch, inlining, and/or polymorphism are used, have their effects on stack usage and timing analysis been addressed?**	2.3.3.1.3	A5: 6

How to Use the OOTiA Handbook

SOI 3 – Verification and Test

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
3.5	**Consider if the testing issues have been addressed by considering the following questions:**		
3.5.1	Has requirements-based testing, software integration testing, and hardware/software integration testing been properly performed? The key concern for requirements testing is that the **mapping of function-oriented test cases to an object-oriented implementation might not be obvious since the basic unit of testing in an OO program is not a function or a subroutine, but an object or a class**.	2.3.3.2.1	A6
3.5.2	Has test coverage of both high-level and low-level requirements been completed? **Note: Test coverage of high-level and low-level requirements typically requires different testing strategies than the traditional functional approach, because information hiding and abstraction techniques decrease or complicate the observability of low-level functions.**	2.3.3.2.1	A7: 3, 4
3.5.3	Has a retest or reuse of test cases approach been implemented to address inheritance and overriding? The concern to be addressed is: **Requirements-based testing is complicated by inheritance, dynamic dispatch, and overriding because it might be difficult to determine how much testing at a superclass level can be reused for its subclasses.**	2.3.3.2.2	A6 & A7

How to Use the OOTiA Handbook

SOI 4 – Final Review

Item #	Evaluation Activity/Question	OOTiA Vol 2 Ref	DO-178B Obj
4.1	Has the approach for addressing OO concerns been summarized in the Software Accomplishment Summary?	N/A	A10: 3
4.2	Have all open OO issues from previous reviews been addressed?	N/A	ALL
4.3	Have all issues identified in the OO Handbook and applicable to the system been assessed and properly addressed?	Vol 1-3	ALL
4.4	Does the Software Configuration Index accurately document the configuration items and environment of the OO software?	N/A	A10:3

How to Use the OOTiA Handbook

Final Thoughts

- Use Volume 2 to identify key concerns and issues you will need to address

- Develop strategies to successfully address the concerns and issues – Volume 3 has some potential solutions

- Evaluate your plan and solutions using Volume 4

How to Use the OOTiA Handbook

Everything you need to know on one slide…

- Volume 1 – Introduction
- Volume 2 – OO things to worry about (either avoid or have an answer)
- Volume 3 – Potential OO answers (not the only answers or all the answers)
- Volume 4 – The kinds of OO questions the certification authorities may ask

Questions?